National College of Ireland

Title

MSc Research Project
Programme Name

Musa Aboki
Student ID: X20218061

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Musa Aboki

Student ID: X20218061

Programme: MSc In Cybersecurity **Year:** 2021-2022

Module: Internship/Research Project

Supervisor: Ross Spelman

Submission Due Date: 1st of February 2023

Project Title: Towards improved phishing detection from URLs, using supervised machine learning

Word Count: 13757 **Page Count** 30

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Musa Idisere Aboki

Date: 1st of February 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Contents

Towards improved phishing detection from URLs, using supervised machine learning

Musa Aboki
X20218061
National College of Ireland, Mayor Street, IFSC, Dublin 1, Ireland.
X20218061@student.ncirl.ie

Abstract

It's still amazing and bemusing how phishing as a social engineering technique has been successfully employed by fraudsters to deceive human users into giving up sensitive and confidential information like email password, bank account numbers or pin numbers, which are later used to defraud individuals and organisations. To combat and defend these forms of attacks often involves organisations using a multi-layered defence mechanism which uses technology solutions that leverages artificial intelligence or machine learning. Defence also involves user awareness training to assist users in knowing the techniques the attackers use. This paper seeks to add to the large and growing body of research to improve the effectiveness of phishing detection by the use of novel machine learning (ML) approach of supervised machine learning classification and algorithms. The Machine learning models used in this paper are Decision Tree-hierarchical structure, Random Forest, Multi-layered Perceptions-Linear (MLP), XGBoost, Autoencoder Neural Network (AE) Support Vector Machines (SVM). The paper would also explore some ensemble approaches to explore if the results can be improved. An ensemble is an ML approach that attempts to improve prediction accuracy through a combination of some of the algorithms mentioned above.

The paper intends to develop a system that uses machine learning techniques to classify websites based on their URLs and tested with datasets from phishing and legitímate URLs. The end result would determine which machine learning model best detects phishing URLs. Suggestions to further improve the results will also be tested and discussed. The paper concludes with additional recommendations for improving detection accuracy.

Keywords
Security, Phishing, Spear phishing, Blacklists, Google Safe Browsing (GSB), PhishTank (PT), OpenPhish (OP), Support Vector Machine, Decision Tree, Random Forest, XGBoost, Classifier based Associative Classification Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Logistic regression (LR), C4.5 generates a decision tree algorithm, Anti-Phishing Working Group (APWG)

1 Introduction

Phishing is "a scalable act of deception whereby impersonation is used to obtain information from a target" that was defined by Lastdrager in (Mourtaji *et al.*, 2021). A known example of phishing attack include spoofing of emails, this starts when phishers send an email to a user by making use of fake email address, which deceives the end user to open the message.(Gunawardena, Kulkarni and Gnanasekaraiyer, 2013). After the user opens the message, the phisher then cajoles the user to get their private information. Simple mail transfer protocol (SMTP) is used in carrying out email spoofing attack. The phisher uses the spoofed email to clone genuine identity of an end user. Spoofing attacks are used to rob data from well-known organisations.(Gunawardena, Kulkarni and Gnanasekaraiyer, 2013)

Cybercrime has now become a menace and it is now one of the most prevalent issues that is affecting businesses globally. It's estimated that by the end of 2021, it spurned a USD 6 trillion annual industry(Morgan, 2022), and is projected to hit $10.5 trillion by 2025. ('Cybercrime Magazine', 2020)

Figure 1: Average total cost and frequency of data breaches by initial attack vector

This is according to the 2021 IBM-Ponemon cybercrime statistics, summarized in the below *Figure 1*
It's also worth a mention that phishing attack was the second most common initial attack vector which was responsible for about 17% of data breaches at an average breach cost of about USD 4.65 million. ('2022 cost of Insider threats Global reports.pdf', no date)

1.1 Phishing and the Analysis of how phishing works

What best describes phishing can be linked to the idea of an individual fishing on a boat. When fishing one would use a piece of food tied to the fishing pole to entice the fish. When the fish bites into the food it's then caught. Likewise, phishing uses the same tactics where cybercriminals make use of the victims' weak points and entice the victims into biting the bait. In the case of phishing, the bait here would be the emails that entice the victims into revealing confidential information by the use of social engineering methods (Sonowal, 2022).
There are four types of phishing described in (*What Are the Different Types of Phishing?*, 2022) namely

- Spear Phishing- These target specific individuals in an organization. e.g., system administrator of the company.
- Whaling- As the name indicates. The target is bigger than a fish, examples would be the company's CEO, CFO within the company, or the business. An email to these individuals would state that the company facing some sort of legal troubles and that they should click on a link provided for more information.
- Vishing- The attack is done via voice calls hence the V rather than the ph. Examples of these would be a call from the attacker representing companies like Microsoft and deceiving victims by informing them they are infected with viruses and asking for victim's credit information to clean up the viruses on the computer.
- Email Phishing-This is the most common type when it comes to phishing attacks. Used since 1990. Emails are sent to victims informing them that their accounts have been hacked and would entice victims to click on a link provided

The means by which attackers make contact with their victims involve emails and SMS. The first step involves sending a link via email and SMS which drives the victim to a fake website. When the victim inputs their personal information, the attacker then initiates contact by using the stolen credentials. (Sonowal, 2022)
Information gathered from phishing attack victims even in small amounts can be of vital importance to phishers. The attackers use the information gathered to produce a seemingly personalized email. The phishers also make it very difficult to catch them by hiding the locations of the servers they use to attack victims.(Vayansky and Kumar, 2018)
Phishing attack typically uses a four-step approach/ process
These are the information (bait), the promise (hook), and the attack (catch). (Woods, 2018)

1. **(Bait)-** A fake malicious website is a setup with the look and feel of the legitimate website page, including down to the web server and DNS server name (Woods, 2018)
2. **(Bait)-** A large number of spoofed emails are sent to target users in the name of the legitimate organization, convincing potential victims to visit the fake website (Woods, 2018)
3. **(Hook/Catch)-** Target victims receive the email, with a tempting offer or urgently required action, open it and click on the embedded malicious website hyperlink and input the required information.
4. **(Catch)-** Cyber criminals steal the personal and/or financial information entered by the unsuspecting victims for nefarious use(Chen and Guo, 2006)

It should also be noted that the term phishing is a play on the word fishing which implies throwing in a net to catch unfortunate victims or (fish).
In the 1990's early hackers used 'ph' to replace 'f' to coin a new word called phishing. The hacker community coined the term because the hacks then were done by telephones. So 'phishing' is a new word formed from 'fishing', it alludes to the deliberate act that the assailant deceives users to visit a faked website by sending them faked messages (or texts), and covertly getting the user' personal details such a, client names, secret keys, public safety IDs, and so forth. These data then can be utilized for future objective commercials or even data fraud assaults (e.g., from the user's bank account). ('History of Phishing', 2019)
Phishing makes use of electronic mail, SMS (SMShing), or voicemail messages. This increased during the era of Covid-19. The attempt relies on the victim's human nature to trust, avarice, fear, and desire to help.

1.2 Evolution of phishing – The inception and beginning of the sophisticated attacks

During the 90s, AOL was one of the main internet service providers and had north of 1,000,000 customers that used their services.
This gigantic fame and geographical spread of AOL got the notice of cyber criminals. A group known as the Warez community, created a malicious software they used on instant messaging, to impersonate AOL employees and trick AOL's subscribers into revealing their passwords through the verification of account or billing information.
AOL being the leading internet service provider with over 1 million subscribers. The scammers then traded the stolen data in AOL chatrooms alongside counterfeit software outlined by (Sonowal, 2022)
As internet websites became widespread, the cybercriminals then graduated to creating fake yahoo and eBay websites/email accounts which had a slight variation on legitimate names, they then used these email accounts to send authentic looking spoofed emails to various users which direct the users to the fake websites and deceive them by extracting billing information.
It has been a tsunami of phishing attacks bonanza since then, which has been launched by cyber criminals using the most sophisticated techniques that range from popup windows to gather sensitive information from unsuspecting users.

1.2.1 Phishing vs Spoofing

In cyber security the term phishing and spoofing usually come up because cybercriminals use phishing as a means of weaponization in the second stage of the cyber kill chain which is using malicious software to exploit a victim's machine. While these terms are different their relationship seems to lead to the usage of a combination of these techniques.
Phishing in contrast to spoofing is discussed in this paper refers to the various tricks in making unsuspecting users perform certain actions which lead to access to personal information. Phishing is primarily a communication / action technique.
A successful phishing expedition on the other hand requires some spoofed tools. Spoofing is an attack where an unknown or untrustworthy form of communication is masqueraded as a legitimate source. Spoofing can then be seen as a subset of most successful phishing attacks.
While phishing is the action, the tools created such as fake emails and fake websites are referred to as spoofs. In other words, the websites used for the completion of a phishing attack are actually spoofed websites. This paper is focused on the website spoofing aspect of a phishing attack.('Spoofing vs Phishing', 2022)
Outlined below is a brief diagram of phishing compared with spoofing in figure 3 below.

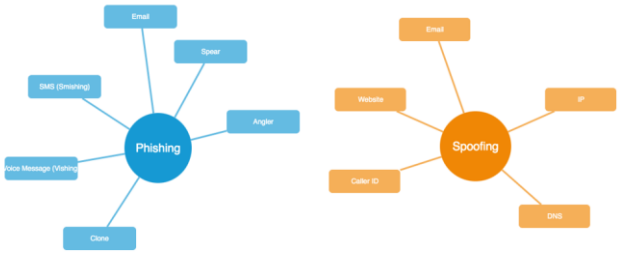


Figure 2 Phishing vs Spoofing

1.3 What motivates these attacks

There are quite a number of motives to carry out a cyber-attack. These include financial gains, insider threats, political motivation (Hacktivism), recognition / achievements and corporate espionage.
Some of these attacks are politically and ideologically motivated outlined by a paper by (Jain and Gupta, 2021)
Cybercriminals often use tactics like phishing attacks to access victims' machines, when they gain access, they modify and encrypt the data, applications, servers and networks or they just infect the entire technical infrastructure of the organisation they have hacked.

1.4 Regulatory Control

In the scope of penalties and key definitions, it states punitive regulatory fines for the victim organization, such as the higher amount of 4% of annual global revenues or 20 million Euros. In the case of personal data breach under the EU General data protection regulation (GDPR), would lead to loss of competitive advantage and business opportunities, or lead to significant reputational damage. Yet victims usually only realise that they have been scammed when their account has already been compromised, or they are contacted by a legitimate technical, fraud or governmental department, or after the cyber criminals have achieved their objective.(*What is GDPR, the EU's new data protection law?*, 2018)

1.5 How to prevent and combat phishing

Defending and prevention of phishing attacks involves a multi-layered approach and also the use of perimeter-based technology solutions such as-:
Perimeter based technology secure web and email gateways, web application firewalls that leverage artificial intelligence and machine learning to detect, inspect and block fake and blacklisted websites, incorrectly spelled email addresses and spot sporadic correspondence designs;
Brand protection services where attackers impersonate an organisations' site. It's advisable to forestall impersonation attacks, where cyber criminals create websites and DNS servers which seem like that of a trusted brand; examples of companies that offer this service is "mimecast"
The implementation of two-factor authentication which goes past the utilization of password validation alone and adds an additional check layer, for example, a token produced on the clients' cell phone, when clients are signing into delicate applications or getting access to sensitive information;
User awareness training to assist clients with recognizing and report assailant strategies and strategies which might sidestep the security perimeter defences.
Most anti-phishing solutions like Avanan, Barracuda Sentinal, BrandShield, suffer from a high rate of false positives and they also lack accuracy. There have been academic studies that focused on the use of data mining algorithms that use sets of features in a paper by (Smadi *et al.*, 2015). Other solutions depend on black-listing and white-listing done in papers but which is not as effective due to the fact that the lifespans of phishing websites are short, at around 2.25 day(SYTNIK and BUBNOV, 2021). Others use content-based approaches with a lexical uniform resource locator (URL)(Zhang, Hong and Cranor, 2007).
Cyber-criminals' methods are becoming more sophisticated and are managing to evade technical controls that companies put in place and are evolving, that is why despite more than 30 years of phishing prevention academic research, phishing has become a profitable business for the cyber criminals. It's estimated that 3 billion emails are sent every day and 156 million of those sent emails bypass perimeter defences resulting in more than 80,000 malicious URLs clicks.(*DocSend - Simple, intelligent, modern content sending*, 2019)
This implies that 5.2% emails are undetected and 0.05% of the undetected emails end up as successful attacks
The objective of this research would be to address, dissect and understand the presence of identifying phishing attacks and use machine learning techniques to identify web-based features like address bar, Domain based features, JavaScript application features and HTML based features involved in phishing attacks.

2 Related Work

Phishing is a well-researched field in academic cycles. There are about the north of 2,000 articles on the definition and theories that outlines the challenges, history and benefits of phishing detection and educate on training approaches. The paper would make an attempt to improve on previous works done and find out the best classification methods towards improved phishing detection from URLs, using machine learning by adding new features and make some feature variations.

2.1 Where it all began to where we are now

Models like logistic regression and neural networks as well as a focus on statistical models was the focus of works of (Quah and Sriganesh, 2008) and (Yue *et al.*, 2007). (Aburrous *et al.*, 2010) created an intelligent system that detected phishing pages in e-banking. Logistic regression and neural networks were based on binary computing models and complex calculations around probability to transform into a straightforward mathematical problem. The model was based on fuzzy logic which is defined as the use of linguistic variables to represent key phishing identities and indicators combined with data mining algorithms derived from an e-banking phishing website. The algorithm achieved 86.38% accuracy based on 27 extracted webpage features. The main challenges of this approach were in determining the right features set that would yield the most optimum results, preparing those features for use and selecting and using the correct data mining technique. Doing this required some intuition from the researchers themselves on the goal of the process of the data mining exercise. The results from the research did however mark the baseline accuracy from which future research would evolve.

By 2011 (Xiang *et al.*, 2011) marked a turning point to the work of phishing detection by the development of CANTINA which was an anomaly-based phishing detection model using Support Vector Machine(SVM) that classified the output. Support Vector Machine (SVM) is a classification and regression predictive tool which makes use of machine learning theory that maximizes predictive accuracy while avoiding the overfit to the data being analysed. (Jakkula, no date)
This page content phishing detection approach was likened to the work of (He *et al.*, 2011). The key features used was the search engine ranking in page classification, which is determined by the rank status of a web page in the search engine i.e., the higher the rank status the legitimisation of the web page. The most significant limitation of this approach was that cyber criminals had a tendency of employing search engine poisoning method to increase the rank status of the malicious website which increases the ranking and hence legitimacy of the website.

In early 2012 and the tail end of 2011(Arade, Bhaskar and Kamat, 2012) made a dive into new algorithm that approximated string matching which compared different webpage addresses in a system. The algorithm comprised of two phases which was URL and Domain identity and second was Image Based Webpage Matching methods. This was done by first using similar URLs with approximate string search algorithm similar to the real URLs stored in a database used by phishers. The I.P addresses were calculated and if they were found as phishing, they were tagged as phishing and transferred to the next detection stage of Image based analysis. In the image-based analysis stage a snapshot of the suspected URL (phishing) and domain identity stage was tagged. When the comparison occurred and it was discovered that above 60% the webpage was tagged as legitimate, otherwise the content of the webpage would be re-checked to determine if the algorithm could be run for all links. The downside of this approach was the increased probability of false positives and incidents, particularly where legitimate webpages were deemed to be phishing pages. However, the phase 2 implementation is in view to address the latency issue with the phase 1 research.

Also in 2012 (Shahriar and Zulkernine, 2012) ran a test on webpages to find out the dependability by the use of a Finite State Machine (FSM) to test the webpage behaviour by tracing the webpage form submission and the corresponding responses. This model is designed to distinguish between phishing websites and legitimate ones based on forms submissions with random inputs. The paper also developed a set of heuristic combinations which captured most recent behaviours of suspected phishing websites. The approach was based on language and the textual content of the individual websites and not based on any updated black or white list. This approach was able to detect advanced XSS-based attacks that a lot of tools in the market fail to detect.

(Ajlouni, Hadi and Alwedyan, 2013) base their research on the build of a 27 feature approach used to train and test the classifiers from (Aburrous *et al.*, 2010) by using the MCAR(an Associative classification (AC) algorithm) while this method resulted in a classification of webpages with a 98.5% accuracy the paper omitted to clarify the amount of rules that were generated by the use of the MCAR algorithm. The ten-fold cross-validation was utilised to evaluate the classification models and produce error rates in the experiments. The 27 features used was not mentioned in the paper.

(Barraclough *et al.*, 2013) in the year 2013 created a new rule-based model which made use of five inputs and employed neuro-fuzzy (termed Fuzzy Logic and Neural Network) model to detect phishing websites. The five input models were made up of tables that stored the features. These included legitimate site rules, user-behaviour profiles, PhishTank, user-specific sites and pop-ups from emails. The model achieved 98.5% phishing attacks detection accuracy. The key drawback of this study was the large number of input features (288 features). which proved both complex and difficult to implement. The feature extraction was based on legitimate site rules where 55 features were extracted, User-behaviour profile where 60 features were extracted that represented user behaviour when using illegitimate sites, Phishtank where 72 features were extracted by using the journal features and 200 phishing websites from the Phishtank archives, User-specific sites extracted 48 features and Pop-Ups from Emails where 42 features were extracted.

(Ramesh, Krishnamurthi and Kumar, 2014) created a model in 2014 that scanned the identity of all direct and indirect links to webpages. The immediate connections separated from the substance of the page and in a roundabout way connected to the removed search engine results. A third-party DNS lookup was also used to map the domains of suspected malicious webpages and the corresponding phishing target IP addresses. This approach achieved 98.62% accuracy but because it has external dependency on 3rd party DNS lookup, the prediction time was dependent upon the speed of the search engine and DNS lookup time. Secondary drawback of the model, was that it could not detect phishing webpages hosted on compromised domains and extract links of keywords from a suspected malicious webpage.

(Abdelhamid, Ayesh and Thabtah, 2014) adopted an approach likened to the rule-based classification in phishing detection where a model proposed was based on summarization of the forecasted error-rate that was produced by a set of Associative Classification (AC) Algorithm used to develop an algorithm called Multi-label Classifier based Associative Classification (MCAC) to extract their rules from training data. The results showed 97.5% classification accuracy which was considered low.

(Zhang *et al.*, 2014) used a five novel feature approach termed as Sequential Minimal Optimization (SMO) algorithm to detect Chinese phishing websites. (Zhang, Yan, Jiang, & Kim, 2014) proposed three out of five features that had never been examined in any prior phishing detection model. The main downside of this model was that the features could only be used on a Chinese webpage.

(Akanbi, Amiri and Fazeldehkordi, 2015) Came up with what is termed as the holistic approach where the paper examined various work and the corresponding countermeasures to determine how to improve the accuracy of phishing detection. First step was focused on dataset gathering, then the dataset was pre-processed which involved reorganising the dataset derived from Phishtank then adding some derived features. After which features were extracted. The features extracted were-: Long URL, Dots-excessive dots in URL, IP address being used instead of registered domains, ssl connection, have @ symbol, hexadecimal, redirection, submit webpage has button included, value assigned to classes.
The second part of the study was based on metric evaluation of the set of classifiers (C4.5, SVM, KNN and LR). The final stage was divided into two parts: the first focused on increased detection rates in phishing website algorithm by selecting the best-suited design for classifier compiler combined with the best ensemble classifier compared with the nest individual classifier. The resulting outcome of the study was 99.37% accuracy which was also attributed to the relatively small size of the dataset.
In the year 2016 (Moghimi and Varjani, 2016) research targeted a new rules-based method which detected phishing attacks in internet banking. The method involved two novel feature sets based on determining webpage identity. Four feature sets were used to test the page resource and identity. And four features were used to determine access protocol of the page element. The relationship between the content and URL of the page was determined by the string-matching algorithm. The feature set was independent from third-party services such as search engines result and web history. The method detected phishing pages in internet banking websites to a 99.14% true positive and 0.86% false positive. The paper employed the method presented by (He *et al.*, 2006) to extract the knowledgebase from the SVM model. The approach combined SVM and decision tree to form a new algorithm called SVM_DT. The main limitation to the model was that the feature set is dependent on webpage content. Detecting webpages with images also proved difficult.

(Zhang *et al.*, 2017) paper was based on the extraction of semantic feature through word2vec which stipulated a better description of the features of phishing sites. They were further fused with multi-scale statistical features to form a robust phishing detection model. The algorithm used AdaBoost, Bagging, Random Forest, and SMO were chosen to implement the learning and testing of phishing detection models., using a simulated environment.

3 Research Methodology

3.1 Research on phishing detection algorithms and considerations

Phishing detection and research have deduced that four key points would improve the accuracy of the phishing detection.

- What machine model is selected
- What features are used and selected to carry out the research
- The sample size used
- The validation and accuracy

The choice of feature selection was based on some previous works done in various studies. Some of the studies where some ideas were adopted include:

- (Akanbi et al 2015). They tried various classification algorithms using a selected set of algorithms similar to the choices used here. A focus on KNN giving a high accuracy was also discussed
- Moghimi and Varjani, 2016). They also used a rule-based approach using 8 feature sets similar to our extraction of feature sets. An ensemble using a decision tree and SVM algorithm proved most accurate. Even though the features used here are different, the approach in checking the correlation between the

different features is also similar. The project would also set up ensembles to check for similar accuracies

- Singh 2020 performed a similar exercise using 11 features. Curiously a single “whois” feature was included checking if registrar name existed or not. Deciding of which features to use is a bit of an art than a science. This choice looks more open for machine learning than making subjective conditions on how long a domain has existed or expiry date being compared to a fixed subjective date.
- It was observed that feature extraction is the backbone of the success of this project. The paper had to repeat this process four times with a need to adjust the extraction algorithm. This was due to a need to adjust the feature set to include the URL length, the number of dots in the URL and the entropy of the URL and domain. Entropy domain is described as when malicious websites insert additional characters in the URL to deceive users to make it look legitimate. Examples can be the likes of MATI which can be written as MAT1 by replacing the letter I with the digit 1. English test has a low entropy which makes it very predictable, the insertion of characters the entropy changes than usual. Malicious URLs were identified by the use of alphabet entropy.(Mamun *et al.*, 2016)

  This was based on a similar study carried out by Mamun et al. titled "Detecting Malicious URLs Using Lexical Analysis.

3.2 Dataset and feature extraction process

The research was carried out in five steps. Namely data gathering, data pre-processing, data transformation, data modelling/data conversion, and result evaluation.(Akanbi, Amiri and Fazeldehkordi, 2015) The process would begin by extracting the features from both the files containing the data set legitimate and phishing URLs. After extraction the features are then split into a training and test sets then the training data sets are tested using the selected classifiers and the results are then used to make predictions on the test sets. All the algorithms used follow these stages except for the autoencoder neural network (AE) due to the semi-supervised approach to learning. Unlike the others, AE training and testing are based on the features only.

The phishing website contains 14306 URLs obtained from the online repository of Phishtank and good URLs from a website owned by University of New Brunswick will be used to get a list of good URLs sites. https://www.unb.ca/cic/datasets/url-2016.html. The page Contains 35378 benign list. This project would be using 16 features with the addition of domain entropy. Efforts would be made to experiment on two sets of feature combination so as to see the impact on accuracy and confusion matrix outcomes. The paper would be sourcing a list of phishing URLs and benign URLs from this site for eventual processing(a) benign from https://www.unb.ca/cic/datasets/url-2016.html and Phishing from Phishtank https://phishtank.org. As of end of May 2022 Alexa.com was retired, so extracting a recent list of page ranks would require unconventional means. Unfortunately, using an existing dataset defeats the need for feature extraction since the Alexa ranking website availability forms a part of the feature extraction procedure.
This dataset contains 15 features extracted from 5000 phishing webpages and 5000 legitimate webpages after adding entropy derived 16 features. These would be labelled as 1 to indicate phishing and 0 to indicate legitimate URLs(Hossain, Sarma and Joyti, 2020) Both phishing and real URLs of websites are gathered to form a dataset and from the required URLs and website content-based features are extracted.

3.2.1 Data Gathering

The data was gathered from two sources. The phishing URLs were gathered from the phish tank website while the benign URLs were gathered from a website owned by University of New Brunswick on https://www.unb.ca/cic/datasets/url-2016.html. The benign database contains 35,378 websites while the phish tank contains 14306 URLs.
The process outlined in the data gathering is outlined below

- Imported the required packages in python (pandas)
- Downloaded the phishing URLs (14306) from Phishtank and benign from the University of New Brunswick which contained URLs gathered from the Alexa.com before it was discontinued in May 2022
- From the combined list 5000 phishing and 5000 benign URLs were extracted and randomized to make up 10,000 URLs

3.2.2 *Date Pre-processing*

The URLs gathered from the websites above contain other extraneous information. This mostly affects the phish tank URL dataset which contains other meta data such as submission time, verified etc. This won't be needed and would be removed before transformation begins. The benign list requires no pre-processing. 10,000 combined results were extracted from both the phishing URLs and the benign URLs in equal proportion.

3.2.3 *Data Transformation / Features Extraction*

The transformation process involved the extraction of 15 features listed in **section 3.4** below using the various functions in **figure 3** also outlined below. Two sets of features would be created to test the differences in accuracy. The first set are based mostly on the work of (Mohammad, Thabtah and McCluskey, 2015) which provides a baseline for comparison. The second set replaced three features based on the lexical approach used by (Darling *et al.*, 2015) where the paper used a lightweight method for classifying malicious URLS at a speed of 0.627 milliseconds and at an accuracy of 99.1%. The J48 classifier was used and compared with the likes of NaïveBayes, BayesianLogisticRegression, LogisticRegression and Knn
This creates a level of input diversity by mixing technology-based features with non-technology-based features. The reason for this was experimental and was focused on combining two research approaches into one to observe if the prediction accuracy would improve positively or negatively and also check how the models identify correlations between these disparate feature types.

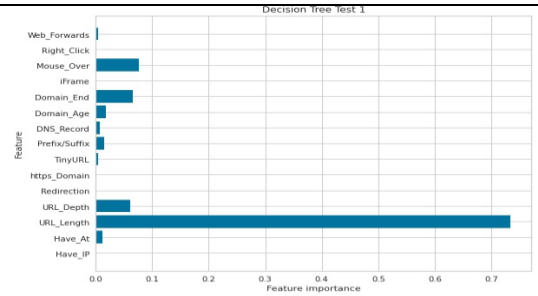
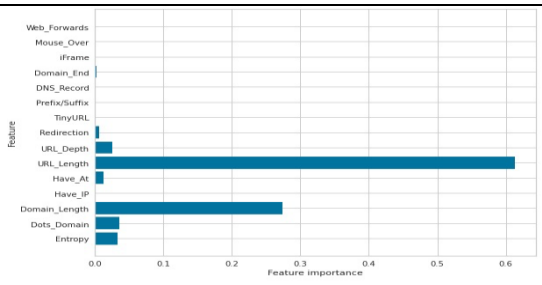
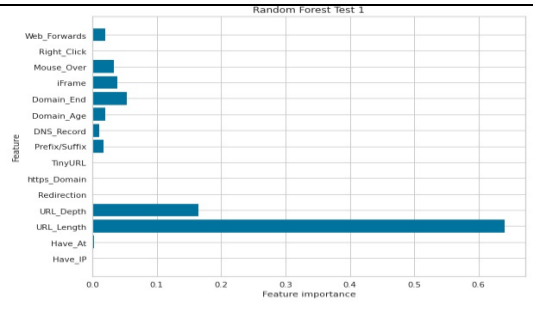
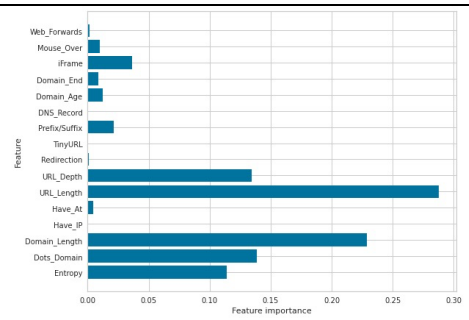
3.2.3.1 *Feature modification and addition of Entropy and dots in the domain*

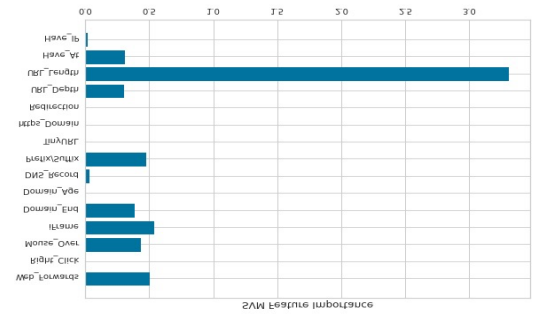
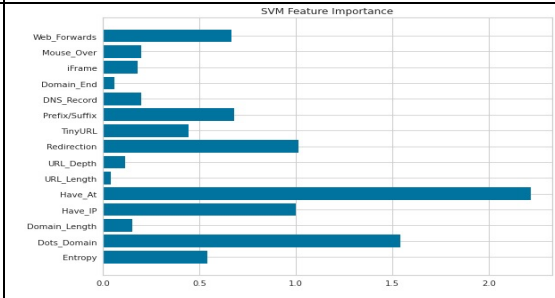
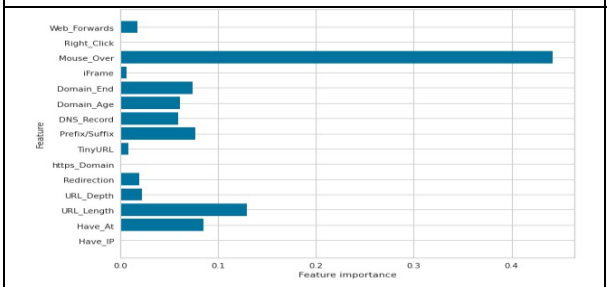
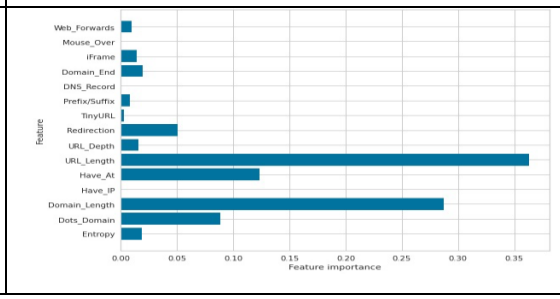
Features extracted into URL data were discovered to require some modifications. This was based on further data inspection and discovery of other domain related features from a lexical perspective. These were entropy and the dots in a domain. Entropy refers to the orderliness of characters in a string. Regular dictionary words have a lower entropy than made up words due to missing syllables and other glaring differences. The entropy calculation is based on the Shannon entropy formula.

3.2.4 *Feature Distribution and Importance for All Classifiers*

The feature importance is dependent on the algorithm within each of the classifiers. As observed, the feature importance differs for decision tree and random forest. This makes arbitrary discard of features with no conviction pointless. That job would be left to the classifier except for obvious cases like missing https etc. However, there are strong correlation changes based on inclusion and exclusion of different features. Notice how the URL length becomes more relevant once the lexical features are included.
Not all classifiers have a way to present importance due to complex weight and adaptation compared to measurement of quality of splits as shown for those below.
The project used Plotting and through this provided a visual view of how the features are distributed across the dataset. This provided an idea on their impact to the machine learning process.

Table 1

Study 1 Done without Entropy Feature	Study 2 Done with Entropy Feature
Decision Tree	
	
Random Forest	
	

SVM	
	
XGBoost	
	

3.2.5 *Data Modelling*

The modelling involves the application of the classifier models on the extracted features. This process includes splitting the 10,000 collated URLs into a training set of 8,000 records and 2,000 record of test sets. The training set is then run through the classifier models and the result is applied on the test set which then measures the accuracy on predictions. All the classifier models follow a similar logic except for the auto encoder which is designed to work on the features only with no reference to the class categories for accuracy testing.

3.3 *The choice of feature selection*

The logic behind using each of these features is explained below
Table 2

	Feature	**Description**
	Domain of URL (Lexical Features)	
1.	Domain Name Entropy	Entropy measures the degree of noise and randomness in a string determines the entropy. Higher string entropy could be suspicious URLs as they don't meet a standard. Entropy is tested on the domain names. So, the higher the entropy the harder it would be to draw any conclusions from the results derived.
2.	Dots in URL	Dots in URL: This is similar to a check on slashes. Malicious URLs reveal quite a number of dots with domain names exhibiting multiple dots due to creation of subdomains. Safe domain names mostly have one dot
3.	Domain Length	Length of Domain: Domain lengths have been shown to have a correlation to malicious and safe websites. This is due to the observation that malicious domain names can be generated using algorithms aptly named Domain Generation Algorithms (DGA). This could lead to unusual domain lengths and a high domain entropy
4.	Have IP Address	IP Address in URL: This is an obvious giveaway on a suspicious website. The ownership is unverified which is most definitely a phishing website. An example could be http://12.345.32.1/signup.html . Phishers are catching up with this. There are only 11 IP address in this dataset
5.	Have "@" Symbol in URL	@ in URL: URLs rarely or if ever have an @ symbol. An @ symbol can be used for many tricks including truncation of preceding values making the conspicuous domain name at the beginning redundant or insignificant to the trues source of information
6.	URL Length	Rather than use a subjective conditional, I would let the machine learn the pattern of URLs
7.	URL Depth (/ in URLs)	Depth of URL: This checks the number of sub-pages in the URL. There are no specific rules or limits here to determine if a site is a phishing site. This relationship is left to the model to determine. The URL depth is included as-is in the feature list. This is the only computed feature with a value that could be greater than 1. There are URLs with depth value as high as 9. It was for this reason I attempted to use feature scaling to see if the results could be much different without scaling. Some previous studies applied the models as-is due to the concept of feature scaling. This concept would be described shortly.
8.	Redirection "//" in URL	Redirection "//" in the URL: The double slash within a URL beyond the protocol (http:// or https://).

		This is a presence of trouble in a URL name. There are 66 redirection markers (//) in our good URL samples and 45 in the phishing URL samples.
9.	"Tiny URL" Using URL Shortening Services	Short URLs: Short URLs can be used even for good URLs but unless absolutely necessary, this is rarely used and if used in useful use cases like reducing character posts on short messaging services like twitter, there counterpart websites also exist. The hiding of the true URL makes it an invaluable took for phishers. Combined with other features, its aides the algorithm in better detecting phising URLs
10.	Prefix or Suffix "-" in Domain	Prefix & Suffix: Websites also rarely contain an hyphen ("-") in their domain names. This makes the presence of a hyphen another worthy marker the detect phishing sites. 70 tiny url services have been included for matching in the URL sample dataset
	Domain Based Features	
11	DNS Record	Check if the DNS record is valid
12.	Domain Age	Age of Domain: These checks how long a domain has existed. Most legitimate domains have existed for at least 6 months. Sites with less than 6 months from creation dates are marked as phishing. This feature is extracted from the whois record using a python library conveniently named whois
13.	Domain End	End Period of Domain: This is a flip on the age of domain. Since, phishing sites are short lived, an active phishing site would be recently registered. Domain names are registered for a minimum period of 1 year. If a domain is legitimate, there are higher chances it wasn't recently created. Most domains have an expiry date less than 6 months. Also note that this condition would catch domain names that were recently created and domains whose registration have been paid in advance for more than a year, however, most domains fall in the less than 6 months category.
	HTML and JavaScript based Features/Content	
14.	Iframe Redirection	Iframe Redirection: iFrames is rarely used in websites. This was popular when websites lacked css stylings and navigation bars and footers were in separate frames within the same web page. The downside is the different frames can be referencing different website URLs compared to the primary URL in the address bar. Since iFrames are rare and are still popular for phishing websites, the presence of an iFrame is a strong indication of a suspicious website
15.	Mouseover	Status Bar Customization: JavaScript provide a client-side immediate response. A lot of mouse movement tricks can easily be applied on a phishing site. Customizing the status bar is one of them. The status bar text can be altered based on the mouse hover within the site which could be misleading to and unobservant user.
16.	Website Forwarding	Website Redirection: Legitimate websites usually have a maximum of one redirection. Phishing sites exhibit multiple redirections.

Features above were extracted by using the Jupyter notebook (python The version of the notebook server is: 6.4.8) The server is running on the version of Python 3.

3.4 Feature Extraction Procedure, Tools & Environment

The performance level of each model is measured and compared. The paper would be experimenting on various supervised learning and one unsupervised learning algorithm as described in section 3.5 titled *Classifier Description and Design*. The feature values are generated through functions parsing the URL and other URL related information like domain information and also website content. The process flow for this algorithm is provided in *figure 3* below. Full details on the approach are in the Jupyter notebook and provided via my ICT Solution Artefact (code, data, etc.) and the configuration manual submitted with the project.

The proposed system consists of using the following classifiers to test the accuracy of detecting phishing URLs. Decision Tree, Multilayer Perception, Random Forest, Autoencoder NN, XG Boost and SVM and results outlined in the *figure 4* below

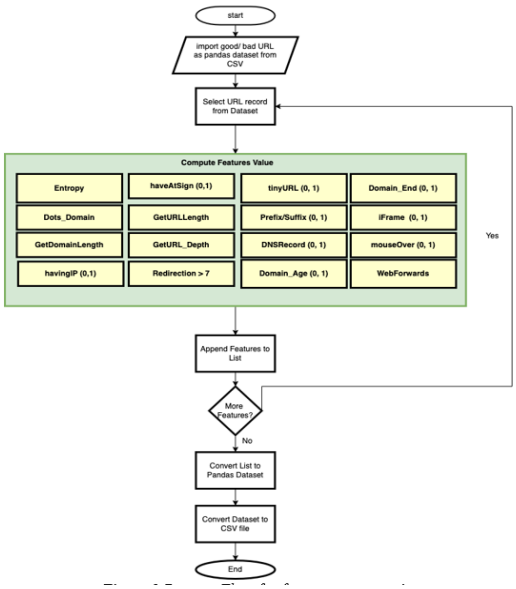


Figure 3 Process Flow for features value creation

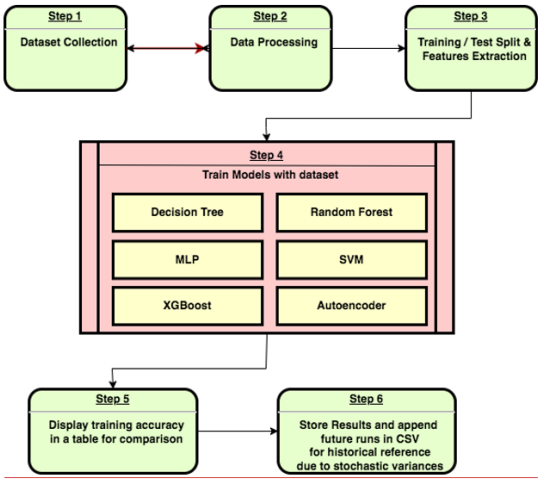


Figure 4 Machine Learning Process Flow

The target and objective of this research project is to prepare AI models and deep neural networks on the dataset made to anticipate phishing sites. Both phishing and harmless URLs of sites are accumulated to frame a dataset and from them is derived a required URL and site content-based features are separated. The output level of each model is measured and analysed.

The project is aimed at using python programming language due to the availability of mature and well tested data management and machine learning libraries such as Scikit, Pandas, NumPy to help analyse and train the dataset. Python has been the preferred programming language due to the availability of these libraries and the ease of using the language. The main objective would be to tag either the websites belonging to phishing or legitimate ones using the machine learning framework. After which the results would be analysed in relation to precision, recall and accuracy. The result would also show the details of the program and how effectively it identifies the websites.
All was done through a simulated environment using the machine learning algorithms within the jupyter notebook installed on a Dell Windows 10 Pro Workstation PC which was moved to the host machine because the virtual environment kept crashing and had issues where the extraction of features was freezing up during the extraction process. All efforts to diagnose the cause of this proved abortive.

These tests were completed with a dataset of 10,000 records equally distributed between the phishing and benign (good) websites.
The feature extraction is carried out through separate Jupyter notebooks for easy of organisation. Once the features were extracted, they were then merged as a single dataset.
This dataset was then run through the various training models and 3 other ensembles. Note that XGBoost and random forest are also forms of ensembles using the decision tree.

3.5 Classifier Description and design

The classifiers used are briefly discussed in relationship to the phishing URL learning process.
The steps are outlined below

3.5.1 Decision Tree

The decision tree is a map of the possible outcomes of a series of related events. The tree's root node commences with the most important feature detected and makes probability decisions branching into nodes of other features and repeats a probability check. It should be pointed out that two tests were carried out on the decision tree classifier. One was on the original features and the other was on a scaled feature set. The scaler removes the mean and scales the values to unit variance. This really was not necessary as most of the features are within the range of 0 and 1 excluding the URL depth.
This was experimental and I was curious to apply this on all the models apart from auto-encoder which seems to use a different approach to learning algorithm.
The results of the training accuracy and training accuracy with scaling added are outlined below.
As this shows that scaling had to effect on the results.

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
1.0	0.993	1.0	0.993

3.5.2 Random Forest

The random forest is an example of ensemble learning. Ensemble learning is a combination of multiple machine learning algorithms to develop a training model. It could be different algorithms or multiples of the same algorithm. The random forest is a combination of multiple decision trees. The dataset (in our case 10,000 records) is randomly split into subsets of the complete URL data. It is like a voting system where the trees with the best result win the learning game. This was the choice of algorithm used by Microsoft to build the Connect gaming system. It is important to point out the parameters tested on the model. Recall, the number of trees was mentioned above. This parameter is referred to as the "n estimator" parameter in the Decision Tree Classifier method. This is defaulted to 100 trees. Running on the default generates an accuracy of 94% for both the training and test sets. Changing from default to a value of 10 surprisingly gives an accuracy of 98%.

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
0.98175	0.9835	0.983625	0.9835

Visualizing Random Forest-The N Estimator of the trees was used to build before taking the maximum vote or average predictions. The higher number of trees derived gives you better performance. Known as the Feynman technique. The Feynman technique is divided into 4 steps; Study, Teach, Fill the Gaps, Simplify.
So the values below compared various n estimators using random values of 1, 10, 7 and 20 to compare the decisions.Called the function and pass the different "I" values into it. Even though it's called k in the function name.

3.5.3 Support Vector Machines (SVM)-

The SVM is popular because it produces significant accuracy with less computing power. It is an improvement on the logistic regression algorithm though quite similar in principle. The algorithm works out the longest dividing line between classes called a hyperplane using the closest points to the centre. This helps prevent future overfitting issues. Visualization of the classification process would be impossible to represent in a multidimensional features dataset. However, a depiction of a two-dimensional (features) dataset is shown below. (Towards Data Science, no date) SVM provided a high accuracy for Moghimi and Varjani, 2016. This would also be tested with the decision tree to test accuracy outcome.

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
0.9625	0.9695	0.96275	0.971

3.5.4 XGBoost

XGBoost is short for eXtreme Gradient boosting. XGBoost is an ensemble learning classifier based on the decision tree classifier. It works through a model feedback mechanism on previously learnt trees. XGBoost was tested and provided the highest accuracy and lowest false negatives and positives in the confusion matrix. This would be shown below. Boosting is a technique that creates a stronger classifier from weaker ones

(James,Witten, Hastie and Tibshirani, 2014). Due to its better classifier regularisation, it tends to prevent over fitting. It also is relatively the fastest of the classifiers used due to the parallel processing of the dataset.
It is highly recommended for structured data as the phishing features are and is the go-to tool of choice in Kaggle competitions. As quoted by the Avito challenge 1st place (Kaggle) winner Owen Zhang, "when in doubt, just use XGBoost"
Predicted the targets value from the model for the samples generated the below results

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
1.0	0.995	1.0	0.995

3.5.5 Multilayer Perceptrons (MLP)

Multilayer perceptron is a deep artificial neural network which comprises of more than one perceptron. They have an input layer which receives the signal and an output layer which makes the decision or the prediction of the input. Also in between those two are a number of hidden layers which involves an engine of computational MLPs (multilayer perceptrons), which a hidden layer capable of calculating any continuous function. In Machine learning MLPs are used to train on a set of input-output pairs and learn to model the degree of dependencies between the inputs and outputs. There is a degree of tweaking involved to train the model and reduce the error rates.(Nicholson, no date)
Predicted the targets value from the model for the samples generated the below results

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
0.991875	0.9885	0.998875	0.9935

3.5.6 Autoencoder

Autoencoder is an unsupervised machine learning that leverages on neural networks for the task of representative learning. The method maps the input of data to internal latent representation which is then used to produce an output. The autoencoder unmasks the underlying diagram of the data by scrutinising the sources of the variant in the input data. The information derived is then used to build the classifiers and other predictors. Developed in 1980s it has since been an important part of neural networks. In the 2000s the autoencoder was used by (Bengio, Lamblin, Popovici, & Larochelle, 2007) to pre-train neural networks. The innovation that gave rise to what is now referred to as "deep learning" was the unsupervised pretraining of neural networks in combination with the development of limited Boltzmann machines (Hinton, Osindero, & Teh, 2006). In this pretraining, the parameters of deep neural networks (DNNs) were started with weight values discovered by autoencoders rather than random values.(Lopez Pinaya *et al.*, 2020)
Steps in the process of training the autoencoder was loaded with all data set and filtered with the good/legitimate URLs. Recall that auto encoder works on encoding the good URLs and then finds a pattern which helps detect URLs that don't meet the signature of the code. An auto encoder is a neural network that has the same number of input neurons as it does outputs. The hidden layers of the neural network will have fewer neurons than the input/output neurons.
Because there are fewer neurons, the auto-encoder must learn to encode the input to the fewer hidden neurons. The predictors (x) and output (y) are exactly the same in an auto encoder.
The below screenshot indicates all the imported programs to the machine learning classifier project
In the autoencoder the procedure are outlined below
Build of the model-Training in Autoencoder
Autoencoder algorith expects the dataset to contain more good outcomes than the bad. It is unreliable to train an imbalanced set of Classes. Even though the project has a balanced set , the scenario would be training the good URLs on the assumption there are inadequate good URLs. The intution is that the encoder would find a pattern within the features that would help it identify badly formed features (phishing URLs)
Once the training test set have been extracted using the train_test_split method, The label would then be dropped from the training set since the encoder wouldn't need it for classification (remember we are using only good URLs so classification would not be necessary). In reverse, we could also carry out this example using the bad URL patterns. That approach should achieve the same result since we are just checking for patterns.
Created a New Dataset and named as X_train_ann ,so we don't corrupt the X_train and X_test values for the supervised learnings as indicated in the screenshot below

3.5.6.1 Build Procedures Below

The model makes use of four fully connected layers with 14,7,7 and 29 neurons. The first two layers are being used for the encoder and the last two used for the decoder. The paper followed similar method used in the article by (Valkov, 2017) The training was done with 100 epochs with a batch size of 32 samples.

3.5.6.2 Evaluating the Results for Autoencoder

The error outlined below on the training and test data seems relatively low as outlined below in the screenshot below.

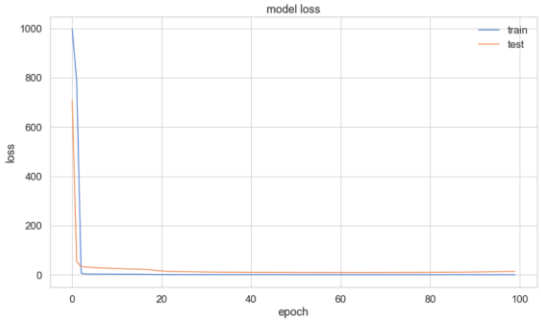


Figure 5 epochs error on training and test data

Error distribution outlined below

```
In [168]:
mse = np.mean(np.power(X_test_ann - predictions, 2), axis=1)
error_df = pd.DataFrame({'reconstruction_error': mse,
                         'true_class': y_test})
error_df.head()
```

Out[168]:

	reconstruction_error	true_class
9394	0.581982	0
898	0.550190	0
2398	5.933540	0
5906	10.002565	1
2343	0.499524	0

```
In [169]:
error_df.describe()
```

Out[169]:

	reconstruction_error	true_class
count	2000.000000	2000.000000
mean	9.329704	0.501500
std	16.764711	0.500123
min	0.345349	0.000000
25%	0.706923	0.000000
50%	0.942494	1.000000
75%	11.535082	1.000000
max	195.015436	1.000000

Figure 6 epochs error distribution

```
fig = plt.figure()
ax = fig.add_subplot(111)
normal_error_df = error_df[(error_df['true_class']== 0) & (error_df['reconstruction_error'] < 10)]
_ = ax.hist(normal_error_df.reconstruction_error.values, bins=10)
```

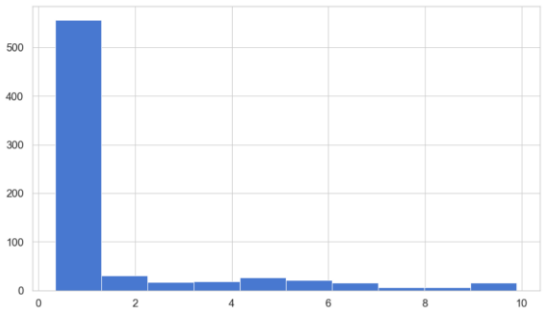


Figure 7 error distribution for the good URLs above

```
In [186]:
fig = plt.figure()
ax = fig.add_subplot(111)
phishing_error_df = error_df[error_df['true_class'] == 1]
_ = ax.hist(phishing_error_df.reconstruction_error.values, bins=10)
```

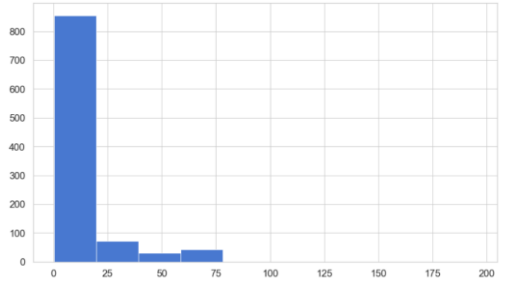


Figure 8 error distribution for the phishing URLs above

The ROC Curve

ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate. False Positive Rate. It graphically depicts the True positives vs the False positives graphically and as outlined below seems to tally and match up okay as screenshot below.

Figure 9 ROC Curve URLs above

Precision and Recall Results measures how many relevant results are returned. Values of 0 and 1 are taken. Ideally a returned result of 1 is needed to be returned. High recall and low precision would mean many results which would mean low and no relevancy. When precision is high and recall is low it's the opposite and few returned results and high relevancy.(Valkov, 2017)

Formulae is

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

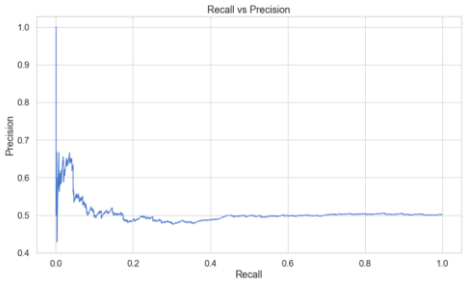


Figure 10 Precision Vs Recall above (Higher area under the curve would mean high recall and high precision)

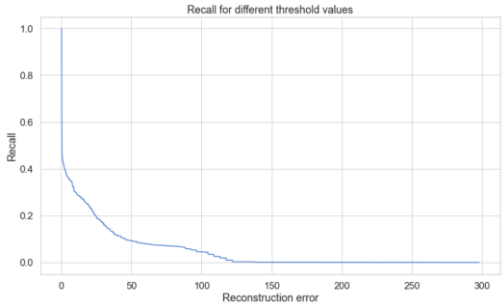


Figure 11 Precision for different threshold values we have the exact opposite situation. As the reconstruction error increases the recall decreases

Prediction: -

In order to predict between phishing and legitimate URLs, we must calculate the reconstruction error from the combined data (ANN (Autoencoder Data). If the error is larger than the threshold, we should conclude it is phishing. We would pick the value as outlined below. This is where Autoencoder becomes tricky as we have the choice of picking our threshold values (sensitivity) arbitrarily) as a rule of thumb and a starting point, the 95% loss as the threshold to identify 5% of data.

Figure 12

3.5.7 KNN- K- Nearest Neighbours Classifier

Defined by (Cunningham and Delany, 2022) as the most straightforward classifier of supervised machine learning techniques. The paper also mentioned the KNN as an important classifier because it issues runtime performance. The KNN is also referred to as the memory-based classification because it is often based in calculations on runtime and referred to as a lazy learning technique. KNN classifies data based on how the neighbours are classified. The algorithm is based on a similarity measure. The K in KNN is based on a parameter which refers to the number of the nearest neighbours that includes the majority of voting process. Due to the algorithm being based on feature similarity making the right choice of the right value for K is called the Parameter tuning and an important factor for better accuracy when the classifier is being calculated.
In the case of the paper, the values chosen are underlined below
The performance evaluation for KNN is screenshot below

Training Accuracy Data	Accuracy Test Data	Training Accuracy Data with Scaling enabled	Test Accuracy Data with Scaling enabled
1.0	0.995	1.0	0.995

3.5.7.1 Performed 10-fold cross validation for KNN

```
from sklearn.model_selection import cross_val_score                          # Use cross_val_score function

neighbors = []                                                               # Empty list to store neighbors
cv_scores = []                                                               # Empty list to store scores

# Perform 10-fold cross validation with K=5 for KNN (the n_neighbors parameter)

for k in range(1, 51, 2):                                                    # Range of K we want to try
    neighbors.append(k)
    knnn = KNeighborsClassifier(n_neighbors = k)                              # k = 5 for KNeighborsClassifier
    scores = cross_val_score(
        knnn, X_train, y_train, cv = 10, scoring = 'accuracy')
    print(k,scores.mean())
    cv_scores.append(scores.mean())

# Passing the entirety of X and y, not X_train or y_train, it takes care of splitting the data
# cv=10 for 10 folds
# Scoring='accuracy' for evaluation metric

scores = cross_val_score(knnn, X_train, y_train, cv=10, scoring='accuracy')
print(scores)
```

Figure 13

[0.97125 0.965 0.96375 0.96 0.96875 0.95875 0.96 0.94625 0.96125 0.97125]

3.5.8 Bagging Ensemble

Bagging helps improve a model's accuracy score while preventing overfitting. Overfitting occurs when a model performs better with the training data and not so great with the test data. Bagging is referred to as bootstrap aggregation, used to create ensemble of decision tree model. Bagging doesn't generally offer an improvement. Bagging was implemented following this details provided by (Brownlee, 2020a) and after implementation the results is screenshot below. In this case the accuracy is outlined below on the datasets. Screenshot below

Decision Tree	0.99225	0.993	
Support Vector Machines	0.960125	0.9695	
Accuracy:	0.994	(0.003)	

3.5.9 Voting Ensemble

There are two types of voting ensembles. These are the soft and hard voting. In classification, a hard voting ensemble involves summing the votes for class labels from other models and predicting the class with the most votes.
A soft voting ensemble involves summing the predicted probabilities for class labels and predicting the class label with the largest sum probability. In other words, Hard is based on the number of classes while Soft is based on probabilities. Details found in the article by (Brownlee, 2020b) and found the hard and soft voting ensemble using the script highlighted below.
The results are outlined below
F1-score of the hard voting classifier: 0.9894
F1-score of the soft voting classifier: 0.995

4 Design Specification

The host machine is a Dell Inspiron Desktop with 500GB hard Disk and running Windows 10 64bit operating system. Database to use would be from Phishtank and for legitimate URLs from the University of New Brunswick https://www.unb.ca/cic/datasets/url-2016.html

Technical Environment Hardware

Physical Device

Device Model	Dell Inspiron Build Machine
Processor	Intel(R) Core (TM) i5-3570 CPU @ 3.40GHz 3.40 GHz
Installed RAM	32.0 GB RAM
System Type	Windows Operating System

Installed Python and Jupyter Notebook to Windows 10 (64 bit) and used the latest version of python which is version 3.10.5

The dataset was pre-processed and resulted in the below outcomes listed below created via a Phishing Data Analysis Python program

URLDATA.CSV	Legitimate.csv	Phishing.csv
10,000 Dataset of pre-processed data of phishing and legitimate URLs	5000 legitimate URLs	5000 phishing URLs

5 Implementation

Two sets of learning tests were carried out on the dataset. The first test had 15 features which included disabling the right click and also availability of https in the URL names (not protocol). It was noticed that the feature extraction found no https and the disabled right clicks were also missing. This meant that they had no importance in the outcome of the training. The accuracy achieved at this point ranged from 91% to 95%. Considering the achievement by Ramesh, Krishnamurthi and Kumar, 2014 and Moghimi and Varjani, 2016, a better result was necessary.

An idea for an improvement test was sought from the work of Darling, Michael 2016 using a lexical approach for identifying the phishing sites. Rather than restart, it was decided that the dropped fields (right click and https) would be replaced with two lexical rules. Since the features had already been extracted with access to the domains, it was decided that this would be the quickest approach to avoid a complete restart. It was quite easy to create functions for the calculation of entropy of the domain and the number of dots in the domain. Phishing sites tend to have more dots than average due to obfuscation and they also have auto generated names with higher entropies than usual. The results generated are then saved into a CSV with future runs appended for comparison. The earlier runs were made and tested about 26 times. The paper suggested we ignore the first 20 as ensembles had not been implemented then. Run 27 was tested with the entropy feature included while run 28 was tested with the two new features. The results were impressively different and the confusion matrices had less false positives and negatives.

6 Evaluation

6.1 Confusion Matrix (CM)

The accuracy of a machine learning algorithm can be misleading. Accuracy measures how many outcome predictions were correct. It doesn't tell which of the classes. Did it get more phishing sites correct or more good URLs? This is where the CM comes in. The system might just miss out on a critical phishing URL which could have a huge cost.

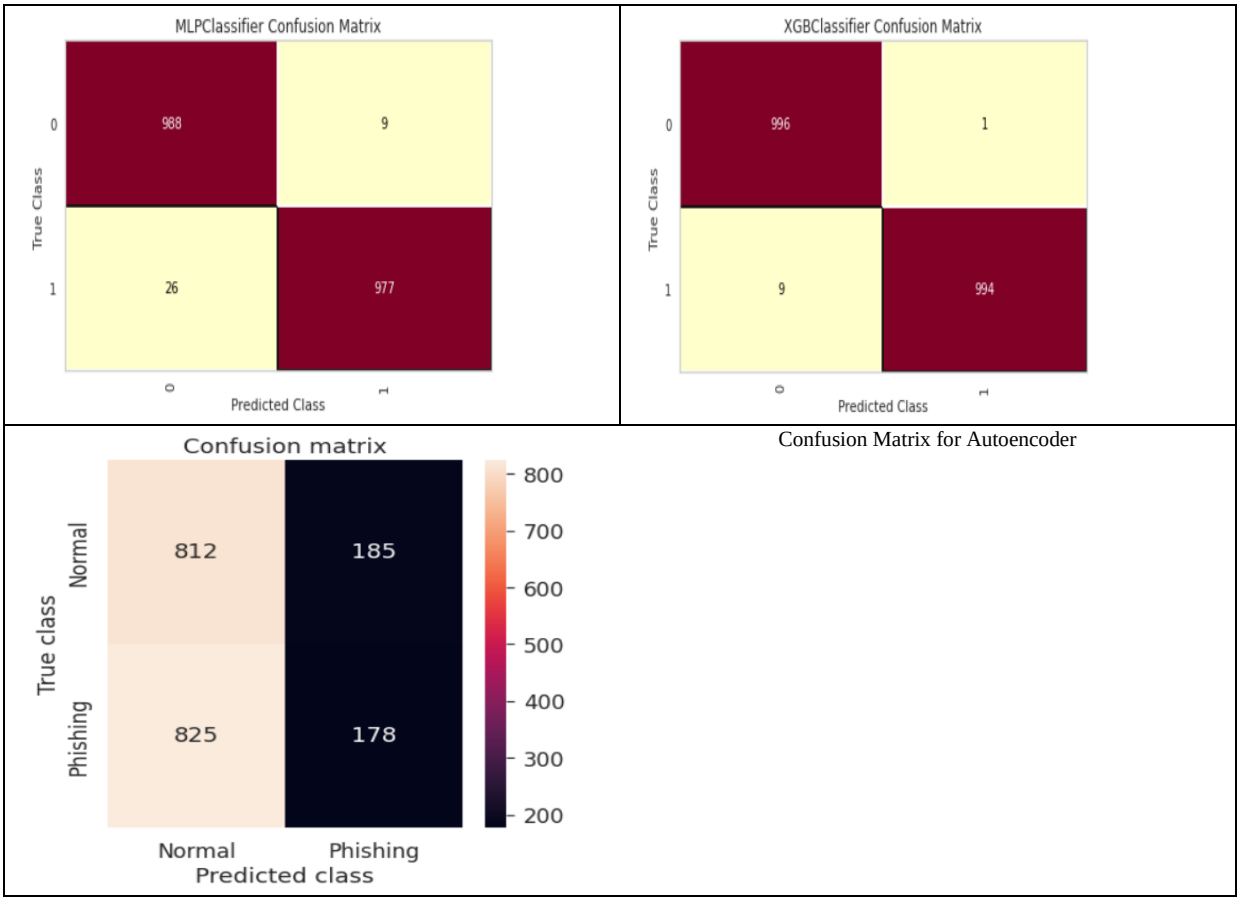
The paper looked at the matrix as measuring the separate accuracy in detecting the phishing URL and good URLs as separate scores.

- True Positives
- True Negatives
- False Positives
- False Negatives

This diagram below clearly explains how the confusion matrix aides in the visual representation of the prediction. The red boxes are correct predictions.

- In the diagram in terms of Decision Tree Classification10 sites were predicted as non-phishing sites but were actually phishing sites while 4 sites were predicted as phishing sites but were actually good sites. Similarly, 993 sites were predicted correctly as non-phishing sites while 993 were predicted correctly as phishing. This is the confusion matrix generated in the jupyter notebook. However, they have been tabulated further down for easy comparison.
- Random Forest Classifier 28 sites were predicted as non-phishing sites but were actually phishing sites while 5 sites were predicted as phishing sites but were actually good sites. Similarly, 992 sites were predicted correctly as non-phishing sites while 975 were predicted correctly as phishing.
- K-Nearest Neighbours (K-NN) Classifier 25 sites were predicted as non-phishing sites but were actually phishing sites while 5 sites were predicted as phishing sites but were actually good sites. Similarly, 992 sites were predicted correctly as non-phishing sites while 978 were predicted correctly as phishing.
- Support Vector Machines (SVM) Classifier 37 sites were predicted as non-phishing sites but were actually phishing sites while 24 sites were predicted as phishing sites but were actually good sites. Similarly, 973 sites were predicted correctly as non-phishing sites while 966 were predicted correctly as phishing.
- Support Vector Machines (SVM) Classifier 37 sites were predicted as non-phishing sites but were actually phishing sites while 24 sites were predicted as phishing sites but were actually good sites. Similarly, 973 sites were predicted correctly as non-phishing sites while 966 were predicted correctly as phishing.
- Multilayer Perceptron Classifier 26 sites were predicted as non-phishing sites but were actually phishing sites while 9 sites were predicted as phishing sites but were actually good sites. Similarly, 988 sites were predicted correctly as non-phishing sites while 977 were predicted correctly as phishing.
- XGBoost Classifier 9 sites were predicted as non-phishing sites but were actually phishing sites while 1 site were predicted as phishing sites but were actually good sites. Similarly, 996 sites were predicted correctly as non-phishing sites while 994 were predicted correctly as phishing. This turned out to be the most accurate
- Autoencoder Neural Network-This needs a bit of tweaking and the results and needs to be studied further as the experiment kept getting different results each time the classifier is generated. 825 sites were predicted as non-phishing sites but were actually phishing sites while 185 site were predicted as phishing sites but were actually good sites. Similarly, 812 sites were predicted correctly as non-phishing sites while 178 were predicted correctly as phishing. Understandably it needs further study.

The Confusion matrix below where the classifiers where ran via the Jupyter notebook



Confusion Matrix for Autoencoder

6.2 The correlation maps

The map shows how related each discrete value are to each other. Lighter boxes show a strong correlation to each other and darker show the opposite. If all the values were strongly correlated across values, then who probably wouldn't need a classification to make serious predictions. However, the classifier makes meaning out of the differences in correlations and find an optimal solution.

Notice the low correlation between URL depth and domain length/ dots in domain and entropy. This feature was one I struggled to make sense of and this shows. The URL lengths are arbitrarily chosen and as long as the depth of the pages chosen. So, a domain could have a URL length as high as 80 or 100 arbitrarily. If this were not arbitrary, there should be a strong correlation between URL length and domain length for example.

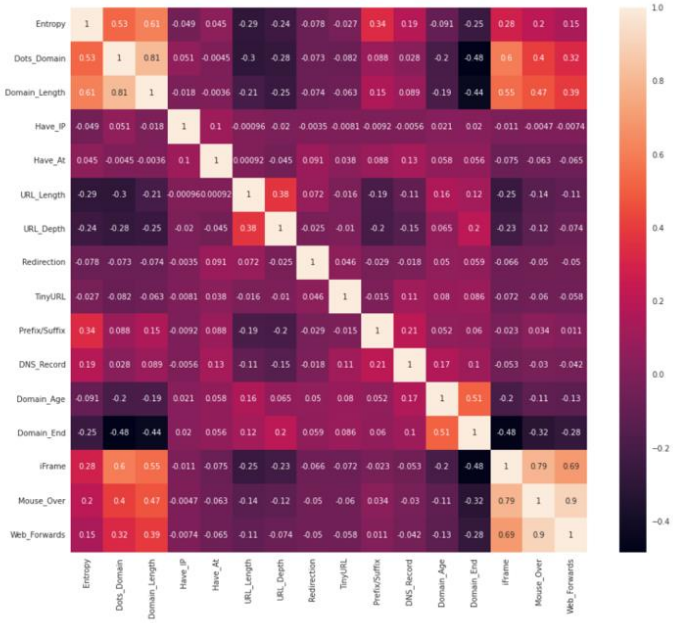
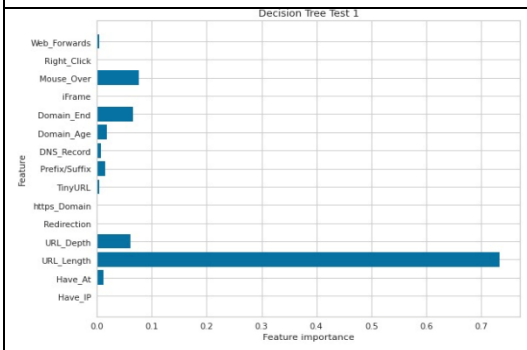
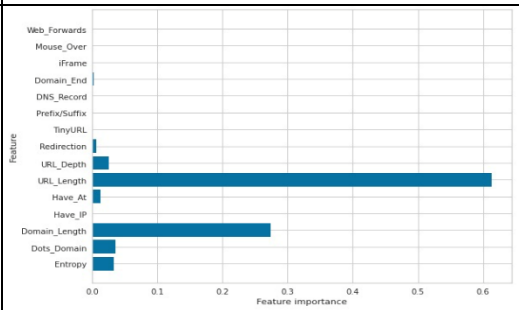
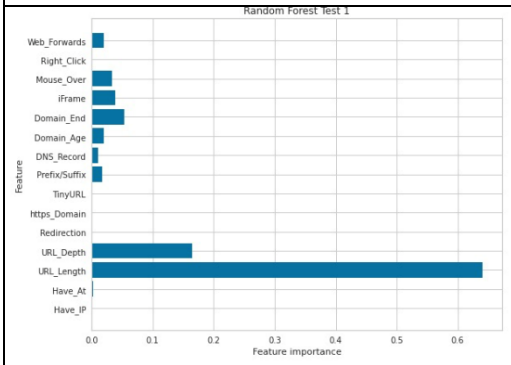
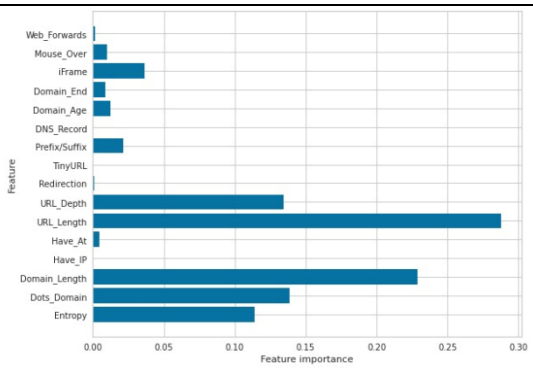
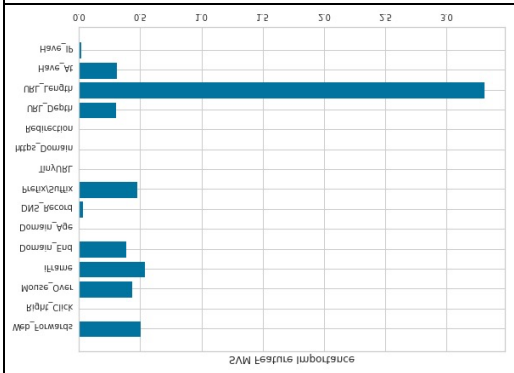
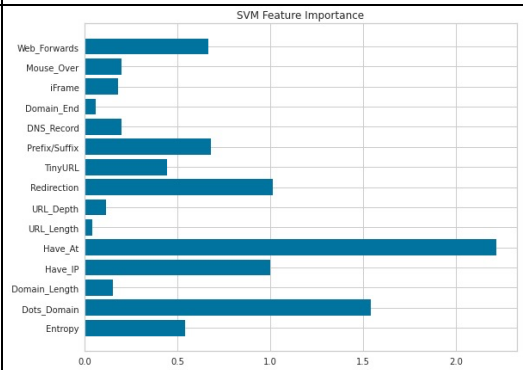
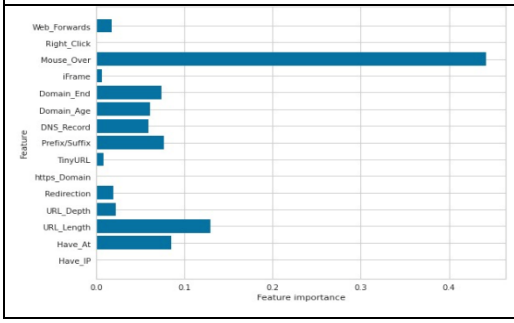
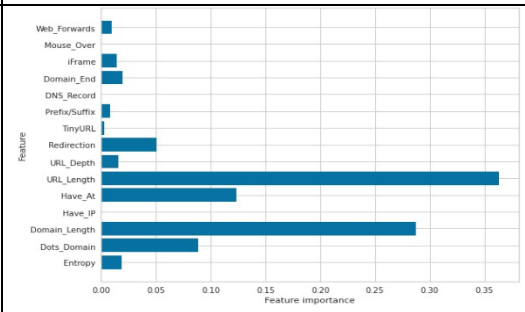


Figure 14 The correlation maps

Experiment / Case Study 1

The initial test was completed without any lexical features applying 15 features. The achieved accuracy seemed reasonable but further checks on the distribution graph revealed excessive over-fitting. The confusion matrix also reveals false negatives as high as 90 and averaging 40. This figure is quite high for detecting phishing sites which could lead to huge financial losses. See table Below with the list of features used in study 1 and study 2

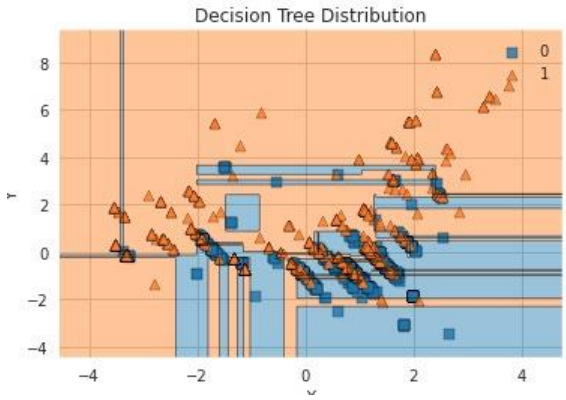
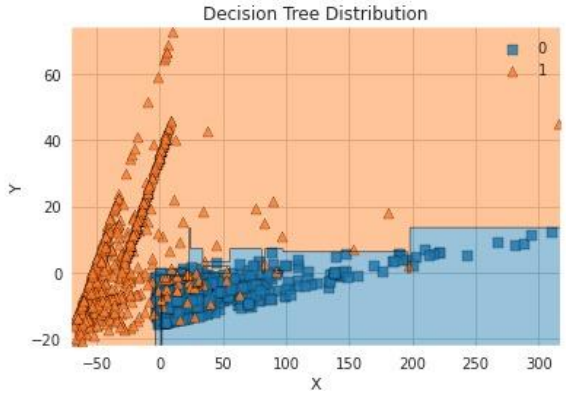
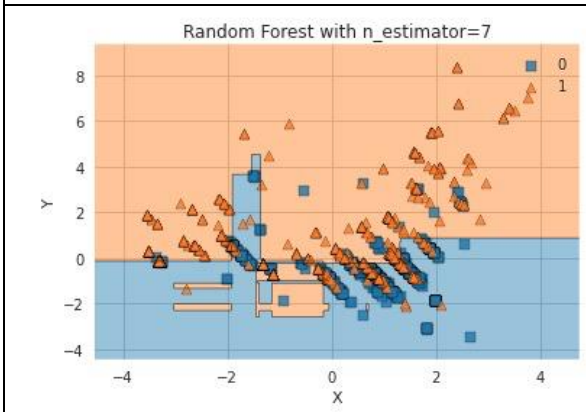
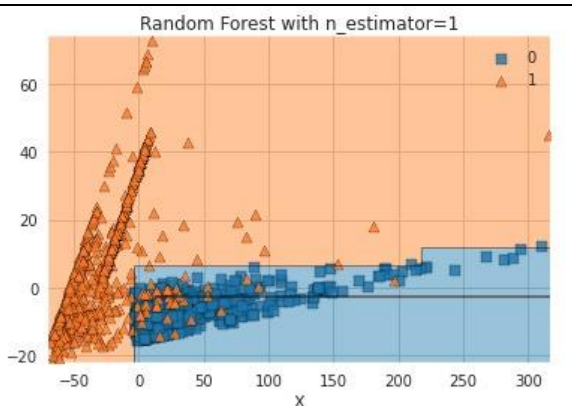
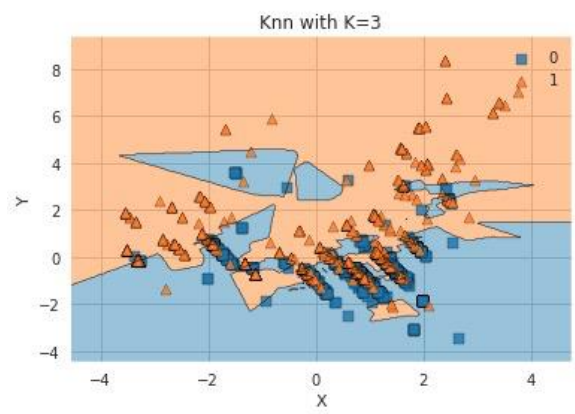
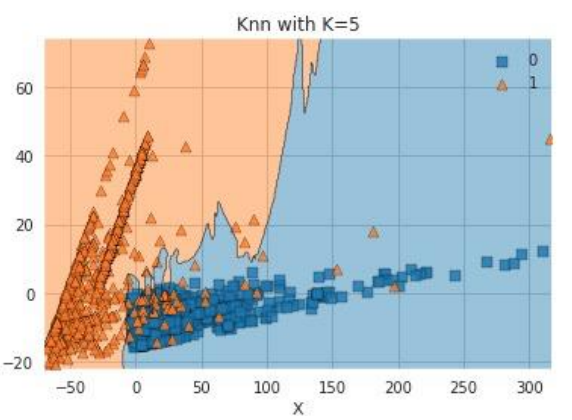
Study 1	Study 2
Decision Tree	
	
Random Forest	
	
SVM	
	
XGBoost	
	

6.3 Discussion

6.3.1 Classification Distribution Graph

The classification distribution is plotted using the Plot decision region method from the mixed library. Plotting the classification is highly recommended to see how the dataset points get plotted and observe overfitting issues. Study 1 shows an incongruency in the plot points and an attempt at overfitting. The SVM was the only classifier that generated a predictable hyperplane.
15 features across different domain. Doman features: 3, Lexical features:9, content features: 4 The result produced accuracies as high as 96 % accuracy for test set on the decision tree. XG Boosts produced similarly high results.
However, the confusion matrix revealed quite a high number of false negatives for both phishing and benign/legitimate sites. The aim of a learning process is to lower the False Negatives as much as possible. Especially for the phishing sites as the financial loss could be huge.
A distribution graph also reveals excessive overfitting on the model to achieve a high accuracy. Curiously, the high percentage accuracy on the test set also conceals the overfitting. Plotting the graphs of the training result

6.4 Experiment / Case Study N

Study 1	Study 2
	
	
	

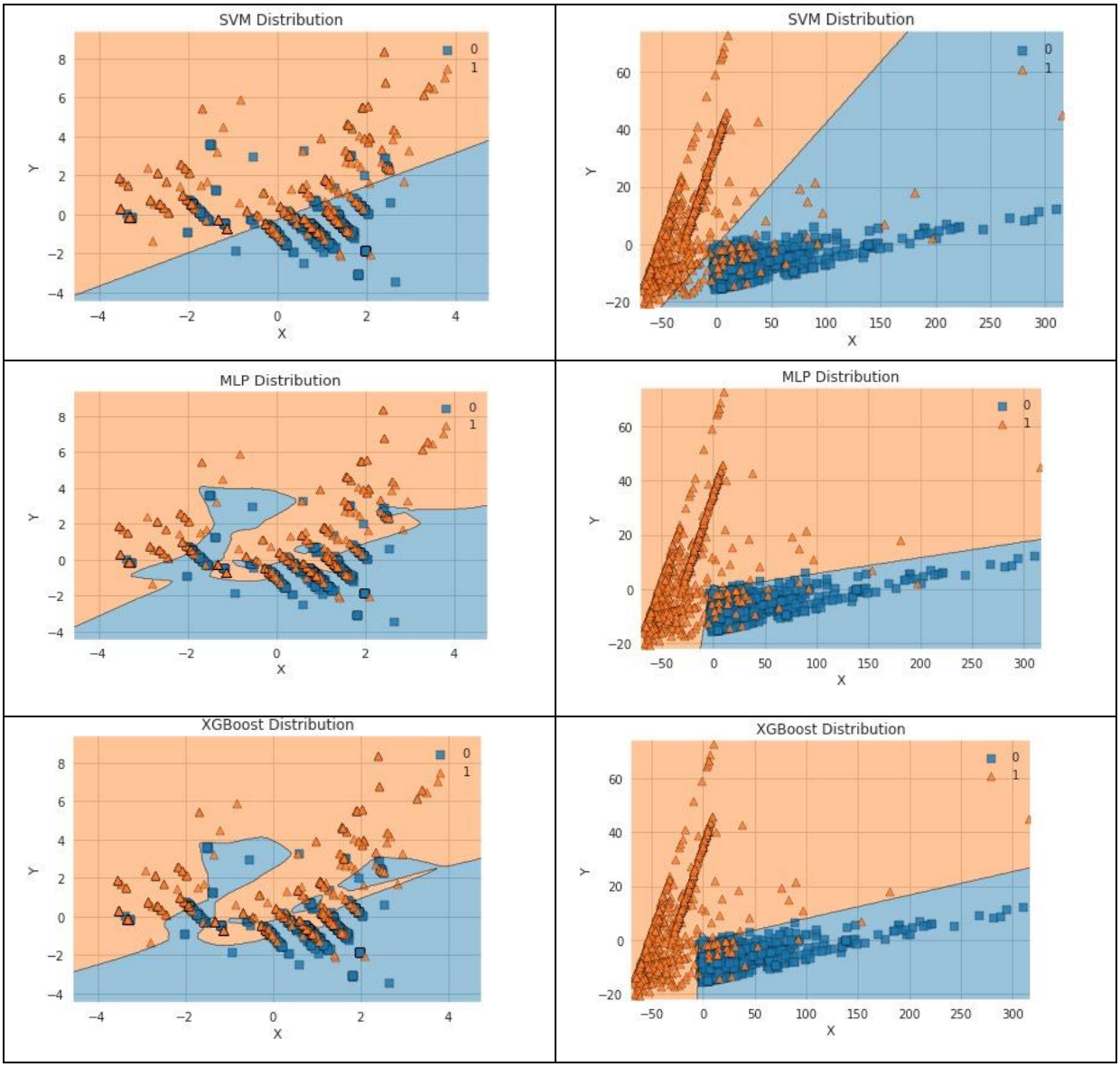


Figure 15 Classification Distribution Graph

6.4.1 Accuracy and Confusion Matrix

Two rounds of studies were completed. One set was based on features which excluded length of the domains and number of dots in the domain. This was based on the study of Akanbi et al. While they got a high accuracy, The experiment couldn't achieve that as there were issues with extracting some domain based and JavaScript based features because we did not use the Alexa base extracted features. This could have been due to unavailability of the Alexa based extracted features as Alexa was discontinued in May 2022.
The second set was adjusted based on the addition of other lexical features as implemented by Mamun et al. 2016. Note that a few addresses based features existed but were not domain-based lexicals.
The lexical approach also solves a problem with retired domains of both phishing and benign websites. These sites would produce the wrong technology-based results (whois, JavaScript content check). However, the historical nature of them having been categorized as phishing and benign makes them a treasure trove of information if their technology-based characteristics is excluded from the feature list. This is why lexical features of the address is the best approach as technology and tricks of phishers keep changing.
Due to the previously completed classification, a choice to combine some lexical features with the existing features and see what the outcome would look like was made for the paper.

Table Confusion Matrix & Test Accuracy

	Accronyms	**Study 1 (without Lexical inclusions)**					**Study 2 (With Lexical Inclusions)**				
	P – Phishing **B – Benign/legitimate** **FN- False Negative** **TP- True Positive**	**TP (P)**	**FN (P)**	**TP (B)**	**FN (B)**	**Accuracy (%)**	**TP (P)**	**FN (P)**	**TP (B)**	**FN (B)**	**Accuracy (%)**
1.	XGB	968	**44**	969	19	96.85	996	**9**	994	1	99.50

2.	Voting (Soft)										99.49
3.	Bagging										99.38
4.	Decision Tree	965	**47**	959	29	96.15	993	**10**	993	4	99.30
5.	Stacking						992	**9**	994	5	96.90
6.	KNN	934	**78**	962	20	95.10	992	**25**	975	5	98.45
7.	MLP	954	**58**	965	23	93.85	983	**16**	987	14	99.45
8.	Random Forest	921	91	972	16	94.65	992	**28**	975	5	98.35
9.	SVM	905	107	931	57	91.95	973	**37**	996	24	96.95

Figure 16 Accuracy & Confusion Matrix Results

Having tested the samples on a combination of 9 classification models. Some were Ensembles which combined various models to achieve a possible different accuracy. The stacking combined the KNN, SVM, Decision Tree and MLP to achieve accuracy results of 99.3%. The bagging combined the SVM and decision tree as done by He eta al (2006) and achieved a combined accuracy result of 99.3

Two sets of datasets were used with a slight variation in features. This was due to an observation that the accuracy computed on the first dataset were lower than those already produced by Ajlouni et al. 2013 (98.5%), Barraclough et al., 2013 (98.5%) and Moghimi and Varjani, 2016 (99.14%).

Two sets of datasets were used with a slight variation in features. This was due to an observation that the accuracy computed on the first dataset were lower than those already produced by Ajlouni et al. 2013 (98.5%), Barraclough et al., 2013 (98.5%) and Moghimi and Varjani, 2016 (99.14%).

Further research revealed a high accuracy from using lexical features of URLs based on works by Mamun et al. 2016 from the university of Brunswick. However, to speed up the process, only two of 48 features were selected in their work to see if the model over-fits less as seen in *study 1* shown in the plot distribution graph in *table 4*. These were dots in domain and domain entropy.

Considering 48 lexical features were used by Mamun et al 2016, increasing the lexical features and excluding all technology related features would surely produce a technology-independent feature set with a high accuracy. The project chose a middle ground based on the traction already achieved using the similar feature sets by Akanbi et al.

The autoencoder was the only unsupervised learning tested. There needs to be a better understanding on this as it follows a completely different principle requiring the feature sets only for training. It requires fixing a threshold value to generate an accuracy. The threshold selection is a combination of an art and a science. I have randomized the threshold as any of the values of the error reconstruction value. Re-running the confusion matrix in the Python Jupyter notebook would regenerate a different outcome every time.

6.4.2 Result Evaluation

The results generated are stored in a CSV file and ordered in descending order with respect to accuracy. The report algorithm was designed to store multiple runs to check the consistency of the results. This also includes storing the confusion matrix as outlined below for the 5 latest results and full results submitted with this paper

	ML Model	Train Accuracy	Test Accuracy	Train Accuracy (Scaled)	Test Accuracy (Scaled)	Confusion Matrix	Execution Time	Run Number
422	AutoEncoder	100.000	100.000	100.000	100.00	[[465, 532], [474, 529]]	Nov 10, 2022, 17:40	52
423	XGB	100.000	99.500	100.000	99.50	[[996 1]\n [9 994]]	Nov 10, 2022, 17:40	52
424	Voting (hard/soft)	98.944	99.499	0.000	0.00	0	Nov 10, 2022, 17:40	52
425	Stacking	100.000	99.400	100.000	99.40	[[994, 3], [9, 994]]	Nov 10, 2022, 17:40	52
426	Decision Tree	100.000	99.300	100.000	99.30	[[993, 4], [10, 993]]	Nov 10, 2022, 17:40	52
427	MLP	99.012	98.850	99.825	99.45	[[992, 5], [18, 985]]	Nov 10, 2022, 17:40	52
428	KNN	99.175	98.500	98.800	98.60	[[992, 5], [25, 978]]	Nov 10, 2022, 17:40	52
429	Random Forest	98.175	98.350	98.362	98.35	[[992, 5], [28, 975]]	Nov 10, 2022, 17:40	52
430	SVM	96.250	96.950	96.275	97.10	[[973, 24], [37, 966]]	Nov 10, 2022, 17:40	52
431	Bagging	99.380	0.000	0.000	0.00	0	Nov 10, 2022, 17:40	52

	ML Model	Train Accuracy	Test Accuracy	Train Accuracy (Scaled)	Test Accuracy (Scaled)	Confusion Matrix	Execution Time	Run Number
0	AutoEncoder	100.000	100.000	100.000	100.00	[[465, 532], [474, 529]]	Nov 10, 2022, 17:40	52
1	XGB	100.000	99.500	100.000	99.50	[[996 1]\n [9 994]]	Nov 10, 2022, 17:40	52
2	Voting (hard/soft)	98.944	99.499	0.000	0.00	0	Nov 10, 2022, 17:40	52
3	Stacking	100.000	99.400	100.000	99.40	[[994, 3], [9, 994]]	Nov 10, 2022, 17:40	52
4	Decision Tree	100.000	99.300	100.000	99.30	[[993, 4], [10, 993]]	Nov 10, 2022, 17:40	52
5	MLP	99.012	98.850	99.825	99.45	[[992, 5], [18, 985]]	Nov 10, 2022, 17:40	52
6	KNN	99.175	98.500	98.800	98.60	[[992, 5], [25, 978]]	Nov 10, 2022, 17:40	52
7	Random Forest	98.175	98.350	98.362	98.35	[[992, 5], [28, 975]]	Nov 10, 2022, 17:40	52
8	SVM	96.250	96.950	96.275	97.10	[[973, 24], [37, 966]]	Nov 10, 2022, 17:40	52
9	Bagging	99.380	0.000	0.000	0.00	0	Nov 10, 2022, 17:40	52

Figure 17 Results of the classifier runs

7 Conclusion and Future Work

The aim of the research was to develop machine learning models to help detect phishing sites from a generated data set. There have been extensive works done in this area by a number of researchers. Some approached the work strictly based on technology-related features while others made an attempt using lexical features.
The paper approached the same problem using a combination of features across both. This was experimental and also an effort to try it across multiple classifiers.
This is usually unnecessary as most researchers usually narrow down to two models or an ensemble combining two models as the case with Amin et al (2019) implementing a decision tree and XGBoost and He et al (2006) who implemented and SVM/ decision tree ensemble. The paper chose and experimented across many classification methods as a learning opportunity into this budding area of machine learning in preparation for future opportunities.
It's also worth a mention that based on the data derived on this research that machine learning has proven to be both a science and an art. The various classifiers fit the science, while getting the right set of features that work seems to be a bit of an art. Though Feature importance can help prune the features further down, getting the wrong set of Features could lead to strange results still. I had to play around with Features to arrive at a meaningful accuracy and most importantly a confusion matrix with a low false positives and negatives.
To test the classifiers, pickle files were created from the training results. This didn't work for all the classifiers but worked for decision tree and random forest classifiers. This wasn't a requirement for the project but a similar test was performed by Anuurag on github using a Flask framework. Part of the code was extracted to build a convenient tool in Jupyter notebooks.
Future works on Phishing would need to focus on areas of application as the algorithm approach has been well researched and implemented. This is why efforts were made to implement a test section using the packaged classifiers as pickle files.
This would also expose future researchers on working towards an End-game on implementation irrespective of the machine learning problems that would be researched.
As mentioned, this exercise on phishing has been highly researched by many data scientists. In the future, I would recommend a focus on existing extracted features dataset so the research can focus on accuracy and confusion matrix outcomes. A deep dive into the Autoencoder unsupervised machine learning model is needed. This was the approach by Aminu et al (2019) who then focused extensively on the performance improvement. However, the drawback on this would be testing of new URLs since no Feature extraction can be applied to new URLs.
The results showed that XGBoost provided the best result and agrees with recommendations as the best tool for structured datasets like in machine learning for phishing detection. This was a completely new field to dig deep into. There was a need to understand the workings of Machine Learning beyond the general understanding and I do understand what this is all about now. I understand what features really do in the real world and the purpose of training and testing data. I also understand various algorithms and the differences in these algorithms. New algorithms would be developed as time goes on and these concepts are now clear to me. I also understand that algorithms can be combined using ensembles to test and research accuracies. I also achieved my expectations in fine-tuning the models and the effects of the addition and removal of features to see the impact on the accuracies achieved. Some of these minor tests would have been omitted from my final write-up due to space constraints but I did see the impact of making minor changes in the feature selection and also parameters used within the learning. Machine Learning is an emerging and critical technology for the future. This was my first practical attempt at it. I felt it made sense as a starter to pick a topic that would be easy for me and others to easily identify with. This also makes me focus on understanding how the model works and how Features connect to the concept of Machine learning.
Phishing as a threat and attack vector in the cyber kill chain has always intrigued me as a topic. So, I chose the topic, towards improved phishing detection from URLs, using supervised machine learning to better understand and test the machine learning models' detection of phishing URLs. It is no joke that the increase in various devices used in web surfing has increased the likelihood of phishing attacks, making it an interesting topic. Machine learning and phishing URLs detection using various algorithms is a verse topic that needed to be understood by myself.

References

‘2022 cost of Insider threats Global reports.pdf’ (no date).

Abdelhamid, N., Ayesh, A. and Thabtah, F. (2014) ‘Phishing detection based Associative Classification data mining’, *Expert Systems with Applications*, 41(13), pp. 5948–5959. Available at: https://doi.org/10.1016/j.eswa.2014.03.019.

Aburrous, M. *et al.* (2010) ‘Intelligent phishing detection system for e-banking using fuzzy data mining’, *Expert Systems with Applications*, 37(12), pp. 7913–7921. Available at: https://doi.org/10.1016/j.eswa.2010.04.044.

Ajlouni, M.I.A., Hadi, W. and Alwedyan, J. (2013) ‘Detecting Phishing Websites Using Associative Classification’, *European Journal of Business and Management*, p. 5.

Akanbi, O.A., Amiri, I.S. and Fazeldehkordi, E. (2015) *A Machine-Learning Approach to Phishing Detection and Defense*. Elsevier. Available at: https://doi.org/10.1016/C2014-0-03762-8.

Arade, M.S., Bhaskar, P.C. and Kamat, R.K. (2012) ‘Antiphishing Model with URL & Image based Webpage Matching’, p. 8.

Barraclough, P.A. *et al.* (2013) ‘Intelligent phishing detection and protection scheme for online transactions’, *Expert Systems with Applications*, 40(11), pp. 4697–4706. Available at: https://doi.org/10.1016/j.eswa.2013.02.009.

Brownlee, J. (2020a) ‘How to Develop a Bagging Ensemble with Python’, *Machine Learning Mastery*, 26 April. Available at: https://machinelearningmastery.com/bagging-ensemble-with-python/ (Accessed: 10 November 2022).

Brownlee, J. (2020b) ‘How to Develop Voting Ensembles With Python’, *Machine Learning Mastery*, 16 April. Available at: https://machinelearningmastery.com/voting-ensembles-with-python/ (Accessed: 10 November 2022).

Chen, J. and Guo, C. (2006) ‘Online Detection and Prevention of Phishing Attacks’, in *2006 First International Conference on Communications and Networking in China*. *2006 First International Conference on Communications and Networking in China*, Beijing, China: IEEE, pp. 1–7. Available at: https://doi.org/10.1109/CHINACOM.2006.344718.

Cunningham, P. and Delany, S.J. (2022) ‘k-Nearest Neighbour Classifiers - A Tutorial’, *ACM Computing Surveys*, 54(6), pp. 1–25. Available at: https://doi.org/10.1145/3459665.

‘Cybercrime Magazine’ (2020) *Cybercrime Magazine*, 10 November. Available at: https://cybersecurityventures.com/cybercrime-damage-costs-10-trillion-by-2025/ (Accessed: 14 August 2022).

Darling, M. *et al.* (2015) ‘A lexical approach for classifying malicious URLs’, in *2015 International Conference on High Performance Computing & Simulation (HPCS)*. *2015 International Conference on High Performance Computing & Simulation (HPCS)*, Amsterdam, Netherlands: IEEE, pp. 195–202. Available at: https://doi.org/10.1109/HPCSim.2015.7237040.

DocSend - Simple, intelligent, modern content sending (2019) *DocSend*. Available at: https://docsend.com (Accessed: 17 July 2022).

Gunawardena, S., Kulkarni, D. and Gnanasekaraiyer, B. (2013) ‘A Steganography-based framework to prevent active attacks during user authentication’, in *2013 8th International Conference on Computer Science & Education*. *2013 8th International Conference on Computer Science & Education (ICCSE)*, Colombo, Sri Lanka: IEEE, pp. 383–388. Available at: https://doi.org/10.1109/ICCSE.2013.6553942.

He, J. *et al.* (2006) ‘Transmembrane segments prediction and understanding using support vector machine and decision tree’, *Expert Systems with Applications*, 30(1), pp. 64–72. Available at: https://doi.org/10.1016/j.eswa.2005.09.045.

He, M. *et al.* (2011) 'An efficient phishing webpage detector', *Expert Systems with Applications*, 38(10), pp. 12018–12027. Available at: https://doi.org/10.1016/j.eswa.2011.01.046.

'History of Phishing' (2019) *PhishProtection.com*, 21 October. Available at: https://www.phishprotection.com/resources/history-of-phishing/ (Accessed: 16 July 2022).

Hossain, S., Sarma, D. and Joyti, R. (2020) 'Machine Learning-Based Phishing Attack Detection', *International Journal of Advanced Computer Science and Applications*, 11(9). Available at: https://doi.org/10.14569/IJACSA.2020.0110945.

Jain, A.K. and Gupta, B.B. (2021) 'A survey of phishing attack techniques, defence mechanisms and open research challenges', *Enterprise Information Systems*, pp. 1–39. Available at: https://doi.org/10.1080/17517575.2021.1896786.

Jakkula, V. (no date) 'Tutorial on Support Vector Machine (SVM)', p. 13.

Lopez Pinaya, W.H. *et al.* (2020) 'Autoencoders', in *Machine Learning*. Elsevier, pp. 193–208. Available at: https://doi.org/10.1016/B978-0-12-815739-8.00011-0.

Mamun, M.S.I. *et al.* (2016) 'Detecting Malicious URLs Using Lexical Analysis', in J. Chen et al. (eds) *Network and System Security*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 467–482. Available at: https://doi.org/10.1007/978-3-319-46298-1_30.

Moghimi, M. and Varjani, A.Y. (2016) 'New rule-based phishing detection method', *Expert Systems with Applications*, 53, pp. 231–242. Available at: https://doi.org/10.1016/j.eswa.2016.01.028.

Mohammad, R.M., Thabtah, F. and McCluskey, L. (2015) 'Tutorial and critical analysis of phishing websites methods', *Computer Science Review*, 17, pp. 1–24. Available at: https://doi.org/10.1016/j.cosrev.2015.04.001.

Morgan, S. (2022) '2022 Cybersecurity Almanac: 100 Facts, Figures, Predictions And Statistics', *Cybercrime Magazine*, 19 January. Available at: https://cybersecurityventures.com/cybersecurity-almanac-2022/ (Accessed: 13 March 2022).

Mourtaji, Y. *et al.* (2021) 'Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network', *Wireless Communications and Mobile Computing*. Edited by C.-H. Chen, 2021, pp. 1–24. Available at: https://doi.org/10.1155/2021/8241104.

Nicholson, C. (no date) *A Beginner's Guide to Multilayer Perceptrons (MLP), Pathmind*. Available at: http://wiki.pathmind.com/multilayer-perceptron (Accessed: 2 November 2022).

Quah, J.T.S. and Sriganesh, M. (2008) 'Real-time credit card fraud detection using computational intelligence', *Expert Systems with Applications*, 35(4), pp. 1721–1732. Available at: https://doi.org/10.1016/j.eswa.2007.08.093.

Ramesh, G., Krishnamurthi, I. and Kumar, K.S.S. (2014) 'An efficacious method for detecting phishing webpages through target domain identification', *Decision Support Systems*, 61, pp. 12–22. Available at: https://doi.org/10.1016/j.dss.2014.01.002.

Shahriar, H. and Zulkernine, M. (2012) 'Mitigating program security vulnerabilities: Approaches and challenges', *ACM Computing Surveys*, 44(3), pp. 1–46. Available at: https://doi.org/10.1145/2187671.2187673.

Smadi, S. *et al.* (2015) 'Detection of phishing emails using data mining algorithms', in *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, Kathmandu, Nepal: IEEE, pp. 1–8. Available at: https://doi.org/10.1109/SKIMA.2015.7399985.

Sonowal, G. (2022) *Phishing and Communication Channels: A Guide to Identifying and Mitigating Phishing Attacks*. Berkeley, CA: Apress. Available at: https://doi.org/10.1007/978-1-4842-7744-7.

'Spoofing vs Phishing' (2022) *Intellipaat Blog*, 25 March. Available at: https://intellipaat.com/blog/spoofing-vs-phishing/ (Accessed: 14 August 2022).

SYTNIK, M. and BUBNOV, E. (2021) *An analysis of the life cycle of phishing and scam pages*, *The life cycle of phishing pages*. Available at: https://securelist.com/phishing-page-life-cycle/105171/ (Accessed: 8 October 2022).

Valkov, V. (2017) *Credit Card Fraud Detection using Autoencoders in Keras — TensorFlow for Hackers (Part VII) | by Venelin Valkov | Medium*. Available at: https://medium.com/@curiousily/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd (Accessed: 26 November 2022).

Vayansky, I. and Kumar, S. (2018) 'Phishing – challenges and solutions', *Computer Fraud & Security*, 2018(1), pp. 15–20. Available at: https://doi.org/10.1016/S1361-3723(18)30007-1.

What Are the Different Types of Phishing? (2022) *Trend Micro*. Available at: https://www.trendmicro.com/en_us/what-is/phishing/types-of-phishing.html (Accessed: 1 October 2022).

What is GDPR, the EU's new data protection law? (2018) *GDPR.eu*. Available at: https://gdpr.eu/what-is-gdpr/ (Accessed: 17 July 2022).

Woods, E. (2018) *The Three Stages Of a Phishing Attack - Bait, Hook And Catch*. Available at: https://blog.usecure.io/three-steps-of-phishing (Accessed: 1 October 2022).

Xiang, G. *et al.* (2011) 'CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites', *ACM Transactions on Information and System Security*, 14(2), pp. 1–28. Available at: https://doi.org/10.1145/2019599.2019606.

Yue, D. *et al.* (2007) 'A Review of Data Mining-Based Financial Fraud Detection Research', in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, Shanghai, China: IEEE, pp. 5514–5517. Available at: https://doi.org/10.1109/WICOM.2007.1352.

Zhang, D. *et al.* (2014) 'A domain-feature enhanced classification model for the detection of Chinese phishing e-Business websites', *Information & Management*, 51(7), pp. 845–853. Available at: https://doi.org/10.1016/j.im.2014.08.003.

Zhang, X. *et al.* (2017) 'Boosting the phishing detection performance by semantic analysis', in *2017 IEEE International Conference on Big Data (Big Data)*. *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA: IEEE, pp. 1063–1070. Available at: https://doi.org/10.1109/BigData.2017.8258030.

Zhang, Y., Hong, J.I. and Cranor, L.F. (2007) 'Cantina: a content-based approach to detecting phishing web sites', in *Proceedings of the 16th international conference on World Wide Web - WWW '07*. *the 16th international conference*, Banff, Alberta, Canada: ACM Press, p. 639. Available at: https://doi.org/10.1145/1242572.1242659.